WHAT IS CLAIMED IS:

1. A method for operating a fault-tolerant server group in client-server distributed dynamic network systems, comprising:

receiving, by a master server in a fault-tolerant server group, a request sent by a client, said fault-tolerant server group comprising said master server and at least one back-up server, said master server registering its mastership in a name server and communicating with both said client and said at least one back-up server, every server in said server group, including said master server and said at least one back-up server, having a self-monitoring mechanism, said self-monitoring mechanism ensuring that said fault-tolerant server group has a consistent mastership situation;

processing, by said fault-tolerant server group, said request to produce a result, said request being processed concurrently by said master server and said at least one back-up server; and

sending, by said master server, said result to said client.

2. The method according to claim 1, further comprising

determining, by said self-monitoring mechanism, whether multiple master servers exist within said fault-tolerant server group; and

restoring a consistent mastership situation in which a sole server serves as said master server in said fault-tolerant server group.

3. A method for operating a self-monitoring mechanism in fault-tolerant distributed dynamic network systems, said method comprising:

detecting an inconsistent situation in which more than a desired number of master servers exist; and

recovering, if said inconsitent stuation is detected by said detecting, from said inconsistent situation to create a consitent situation in which the desired number of master server exists.

4. The method according to claim 3, wherein said detecting an inconsistent situation comprises identifying:

a master server that is not a name server master server, wherein said name server master server is a server defined as a master in a name server, said master server that is different from a name server master server causing said inconsisitent situation; or

a master of a back-up server that is not a name server master server, the master of said back-up server causing said inconsistent situation.

5. The method according to claim 4, wherein said identifying a master server comprises:

selecting a server whose state indicates that said server is a master;

determining said server, selected by said selecting, as said master server that causes said inconsistent situation if said server is not a name server master server defined in said name server; and

setting the state of said server as master if said server is the name server master server.

6. The method accoring to claim 4, wherein said identifying a master of a back-up server comprises:

selecting a server whose state indicates that said server is a back-up; and

determining the master of said server, selected by said selecting, as causing said inconsistent situation if the master of said server is not a name server master server defined in said name server.

7. The method according to claim 4, wherein said recoverying from said inconsistent situation comprises:

setting the master of a server, identified by either said identifying a master server or said identifying a master of a back-up server, to be a name server master server;

synchronizing the state of said server with the state of said name server master server;

terminating said server if said synchronizing is not successful; and

setting the state of said server as a back-up, if said synchronizing is successful.

8. The method according to claim 7, wherein said synchronizing comprises:

downloading the state of a name server master server from said name server master server to said server; and

aligning the state of said server with said state of the name server master server, downloaded from said name server master server.

9. The method according to claim 7, further comprising:

comparing, if said synchronizing is successful, the priority of said server with the priority of said name server master server, said priority being defined according to at least one criterion; and

setting the state of said server as a master, if the priority of said state is higher than the priority of said name server master server.

10. The method according to claim 9, wherein

said at least one criterion includes at least one of computing speed and capacity.

11. The method according to claim 4, further comprising:

initializing the state of said server;

initializing a time-out condition;

setting up a timer; and

performing said detecting when said timer achieves said time-out condition.

12. The method according to claim 3, further comprising triggering a server to perform said detecting.

13. The method according to claim 12, wherein

said triggering includes triggering using a time-out mechanism;

said triggering includes triggering by a master server; and

said triggering includes triggering by a name server when said name server detects multiple server IDs that correspond to a same name.

14. The method according to claim 13, wherein said triggering using a time-out mechanism includes triggering using a time-out mechanism based on a timer.

15. The method according to claim 13, wherein said triggering by a master server includes triggering when said master server is an original name server master server and when there is at least one different master server, registered in said name server that are not an original name server master server.

16. The method according to claim 3, further comprising reinitializing a time-out mechanism when no inconsistent situation is detected by said detecting.

17. A method for operating a name server, said method comprising:

detecting multiple registrations of master servers; and

retaining, when multiple registrations of master servers are detected, one master server registration according to a criterion.

18. The method according to claim 17, wherein said multiple registrations of master servers use a same server group's name with different server IDs.

19. The method according to claim 18, wherein said retaining includes keeping a registration of a master server that has the lowest server ID.

20. The method according to claim 17, further comprising:

triggering a self-monitoring mechanism when multiple registrations of master servers are detected.

21. A fault-tolerant server group in distributed dynamic network systems, comprising:

a client;

a fault-tolerant server group for providing a service to said client, said fault-tolerant server group comprising at least one master server and at least one back-up server, said master server communicating with said client, said fault-tolerant server group having a self-monitoring mechanism that ensures that a consistent mastership situation in said fault-tolerant server group; and

a name server for registering the mastership of a master server corresponding to said fault-tolerant server group.

22. The system according to claim 21, wherein said self-monitoring mechanism includes a portion installed on said at least one master server and said at least one back-up server, in said fault-tolerant server group.

23. A self-monitoring mechanism in fault-tolerant distributed dynamic network systems, comprising:

a detection mechanism for detecting an inconsistent situation in which more than a desired number of master servers exist; and

a recovery mechanism for recoverying, if said inconsistent situation is detected by said detection mechanism, from said inconsistent situation to create a consistent situation in which said desired number of master servers exist.

24. The system according to claim 23, wherein said detection mechanism comprises:

a trigger that reacts upon an external event to activate said detection mechanism to perform said detecting;

a time-out mechanism for generating an activation signal, according to a time-out criterion, to start said detecting; and

a detector for performing said detecting, said detector being activated by either said trigger or said time-out mechanism.

25. The system according to claim 24, wherein said external event includes when a name server detects more than the desired number of registrations of master servers.

26. The system according to claim 24, wherein said external event includes when a master server detects the existence of another master server.

27. The system according to claim 24, wherein said time-out mechanism includes a timer and counts towards said time-out criterion based on said timer.

28. The system according to claim 24, wherein said detector comprises:

an initializer for initializing a timer, time-out criterion, and self-monitoring state;

a determiner for determining whether a server is involved in said inconsistent situation.

29. The system according to claim 23, wherein said recovery mechanism comprises:

an alignment mechanism for aligning a server with said master server by assigning one of said master servers as the master of said server;

a synchronization mechanism for synchronizing the state of said server with the state of said one of said master servers; and

a state assignment mechanism for assigning the state of said server.

30. A system of a name server, said system comprising:

a detector for detecting multiple registrations of master servers; and

a correction unit for, when multiple registrations of master servers are detected, retaining only one master server registration.

31. The system according to claim 30, further comprising:

a triggering mechanism for triggering said self-monitoring when multiple registrations of master servers are detected.

32. A computer readable medium having program code stored thereon, such that when the code is read and executed by a computer, the computer is caused to:

receive, by a master server in a fault-tolerant server group, a request sent by a client, said fault-tolerant server group comprising said master server and at least one back-up server, said master server registering its mastership in a name server and communicating with both said client and said at least one back-up server, every server in said server group, including said master server and said at least one back-up server, having a self-monitoring mechanism, said self-monitoring mechanism ensuring that said fault-tolerant server group has a consistent mastership situation;

process, by said fault-tolerant server group, said request to produce a result, said request being processed concurrently by said master server and said at least one back-up server; and

send, by said master server, said result to said client.

33. The medium according to claim 32, wherein the code recorded on the medium further causes the computer to:

determine, by said self-monitoring mechanism, whether multiple master servers exist within said fault-tolerant server group; and

restore a consistent mastership situation in which a sole server serves as said master server in said fault-tolerant server group.

34. A computer readable medium having program code stored thereon, such that when the code is read and executed by a computer, the computer is caused to:

detect an inconsistent situation in which more than a desired number of master servers exist; and

recover, if said inconsitent stuation is detected by said detecting, from said inconsistent situation to create a consitent situation in which the desired number of master server exists.

35. The medium according to claim 34, wherein the code recorded on the medium further causes the computer to identify:

a master server that is not a name server master server, wherein said name server master server is a server defined as a master in a name server, said master server that is different from a name server master server causing said inconsisitent situation; or

a master of a back-up server that is not a name server master server, the master of said back-up server causing said inconsistent situation.

36. The medium according to claim 35, wherein the code recorded on the medium further causes the computer to perform said identifying of a master server by:

selecting a server whose state indicates that said server is a master;

determining said server, selected by said selecting, as said master server that causes said inconsistent situation if said server is not a name server master server defined in said name server; and

setting the state of said server as master if said server is the name server master server.

37. The medium according to claim 35, wherein the code recorded on the medium further causes the computer to perform said identifying of a master of a back-up server by:

selecting a server whose state indicates that said server is a back-up; and

determining the master of said server, selected by said selecting, as causing said inconsistent situation if the master of said server is not a name server master server defined in said name server.

38. The medium according to claim 35, wherein the code recorded on the medium further causes the computer to:

set the master of a server, identified by either said identifying a master server or said identifying a master of a back-up server, to be the name server master server;

synchronize the state of said server with the state of said name server master server;

terminate said server if said synchronize is not successful; and

set the state of said server as a back-up, if said synchronize is successful.

39. The medium according to claim 38, wherein the code recorded on the medium further causes the computer to:

download the state of a name server master server from said name server master server to said server; and

align the state of said server with said state of the name server master server, downloaded from said name server master server.

40. The medium according to claim 38, wherein the code recorded on the medium further causes the computer to:

compare, if said synchronize is successful, the priority of said server with the priority of said name server master server, said priority being defined according to at least one criterion; and

set the state of said server as a master, if the priority of said state is higher than the priority of said name server master server.

41. The medium according to claim 35, wherein the code recorded on the medium further causes the computer to:

initialize the state of said server;

initialize a time-out condition;

set up a timer; and

performing said detecting when said timer achieves said time-out condition.

42. The medium according to claim 34, wherein the code recorded on the medium further causes the computer to trigger a server to perform said detect.

43. The medium according to claim 42, wherein

said trigger includes triggering using a time-out mechanism;

said trigger includes triggering by a master server; and

said trigger includes triggering by a name server when said name server detects multiple server IDs that correspond to a same name.

44. The medium according to claim 43, wherein said trigger using a time-out mechanism includes a trigger using a time-out mechanism based on a timer.

45. The medium according to claim 43, wherein said triggering by a master server includes triggering when said master server is an original name server master server and when there is at least one different master server, registered in said name server that is not the original name server master server.

46. The medium according to claim 34, wherein the code recorded on the medium further causes the computer to reinitialize a time-out mechanism when no inconsistent situation is detected by said detect.

47. A computer readable medium having program code stored thereon, such that when the code is read and executed by a computer, the computer is caused to:

detect multiple registrations of master servers; and

retain, when multiple registrations of master servers are detected, one master server registration according to a criterion.

48. The medium according to claim 47, wherein said multiple registrations of master servers use a same server group's name with different server IDs.

49. The medium according to claim 47, wherein said retaining includes keeping a registration of a master server that has the lowest server ID.

50. The medium according to claim 47, wherein the code recorded on the medium further causes the computer to:

trigger a self-monitoring mechanism when multiple registrations of master servers are detected.